Countable Sets

Infinite sets are either:

- Countable
- Uncountable

Countable set:

There is a one to one correspondence

between

elements of the set

and

positive integers

Example: The set of even integers is countable

Even integers: $0, \; 2, \; 4, \; 6, \; \ldots$

Correspondence: $\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

Positive integers: $1, \; 2, \; 3, \; 4, \; \ldots$

$2n$ corresponds to $n+1$

Example: The set of rational numbers is countable

Rational numbers: $\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$

Naive Approach

Rational numbers: $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \ldots$

Correspondence: $\downarrow \quad \downarrow \quad \downarrow$

Positive integers: $1, \; 2, \; 3, \; \ldots$

Doesn’t work:

we will never count numbers with nominator 2

$$\frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \ldots$$

Better Approach

$$\frac{1}{1} \quad \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{4} \quad \cdots$$

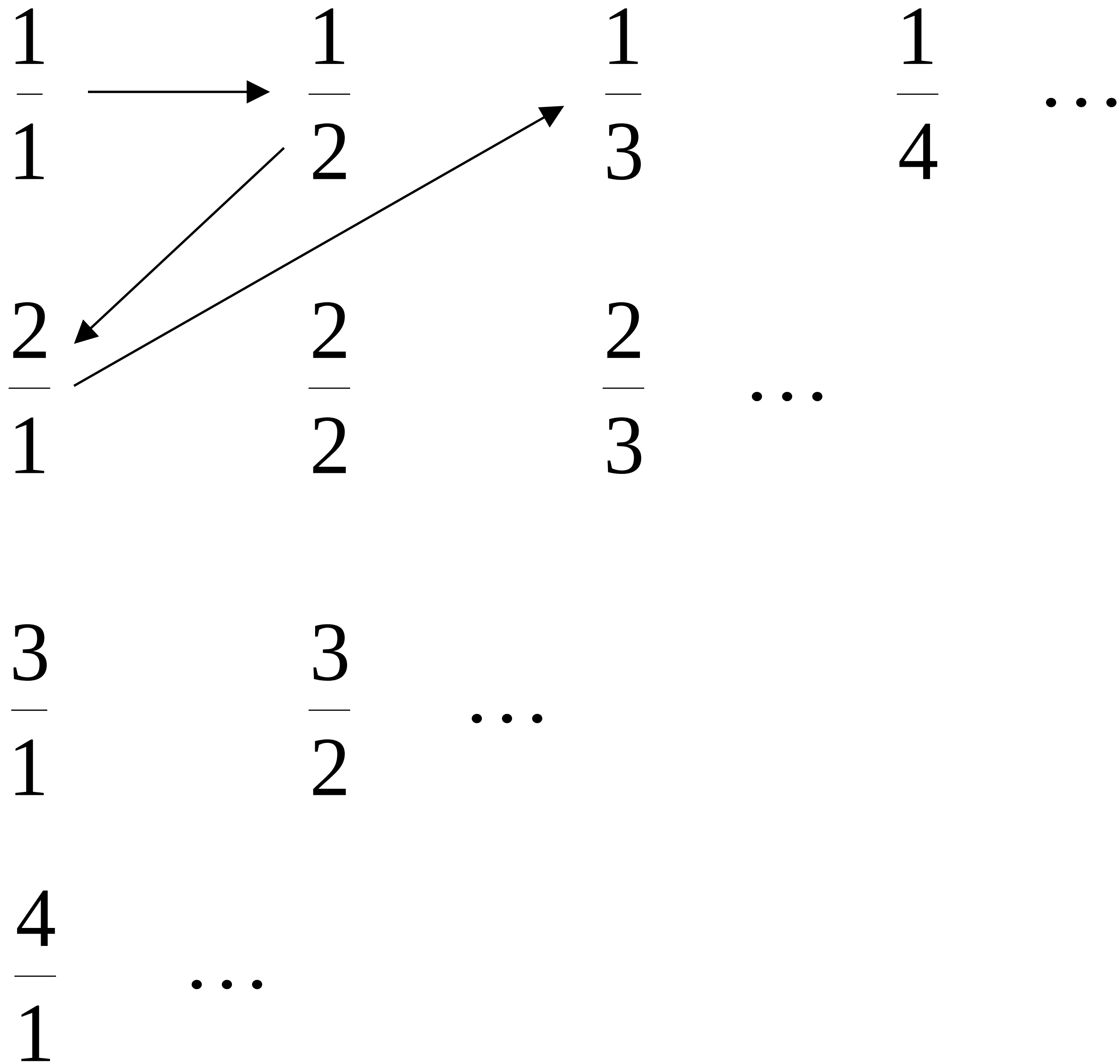
$$\frac{2}{1} \quad \frac{2}{2} \quad \frac{2}{3} \quad \cdots$$

$$\frac{3}{1} \quad \frac{3}{2} \quad \cdots$$

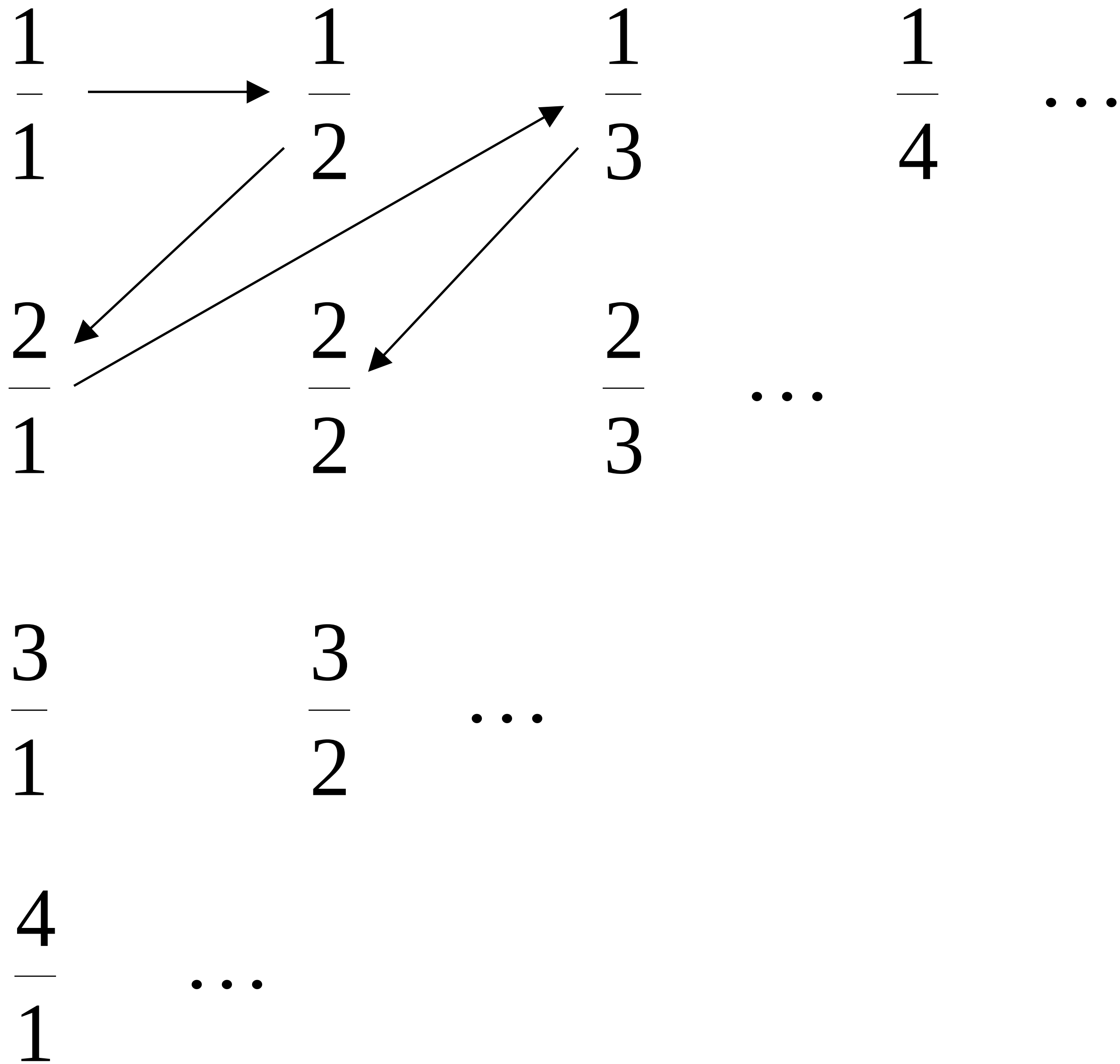
$$\frac{4}{1} \quad \cdots$$

$$
\begin{array}{ccccc}
\frac{1}{1} \longrightarrow & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}
$$

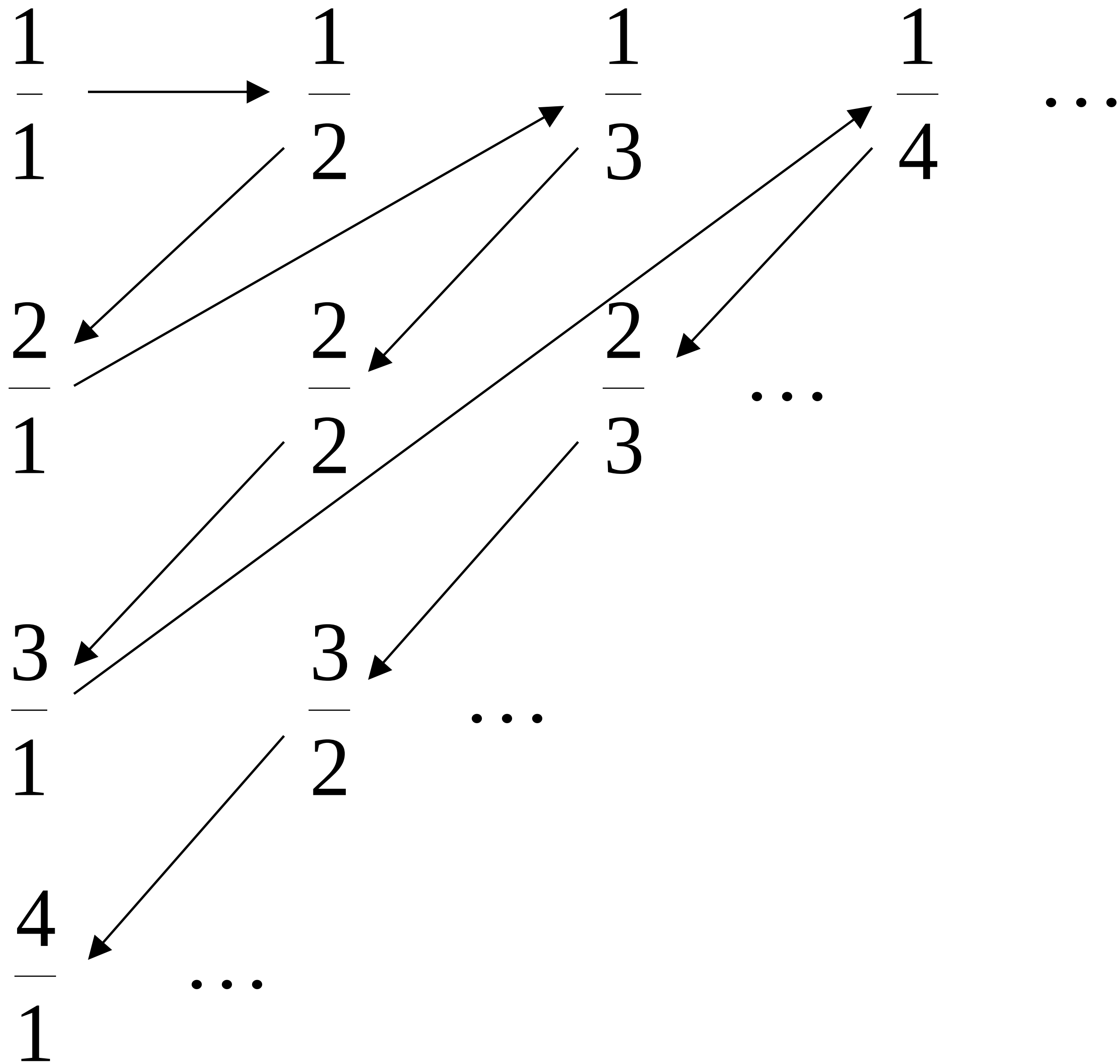
$$
\begin{array}{ccccccc}
\frac{1}{1} & \rightarrow & \frac{1}{2} & & \frac{1}{3} & & \frac{1}{4} & \cdots \\
& \swarrow & & & & & \\
\frac{2}{1} & & \frac{2}{2} & & \frac{2}{3} & \cdots & \\
& & & & & & \\
\frac{3}{1} & & \frac{3}{2} & \cdots & & & \\
& & & & & & \\
\frac{4}{1} & \cdots & & & & &
\end{array}
$$



1/1 1/2 1/3 1/4 ...
2/1 2/2 2/3 ...
3/1 3/2 ...
4/1 ...



1/1
1/2
1/3
1/4
…
2/1
2/2
2/3
…
3/1
3/2
…
4/1
…


1/1
1/2
1/3
1/4
. . .
2/1
2/2
2/3
. . .
3/1
3/2
. . .
4/1
. . .

Rational Numbers: $\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \ldots$

Correspondence: $\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

Positive Integers: $1, \; 2, \; 3, \; 4, \; 5, \; \ldots$

We proved:

the set of rational numbers is countable

by giving

an **enumeration procedure**

Definition

Let S be a set of strings

An **enumeration procedure** for S is a Turing machine that generates any string of S in finite number of steps.

strings $s_1, s_2, s_3, \ldots \in S$

Enumeration Machine for S

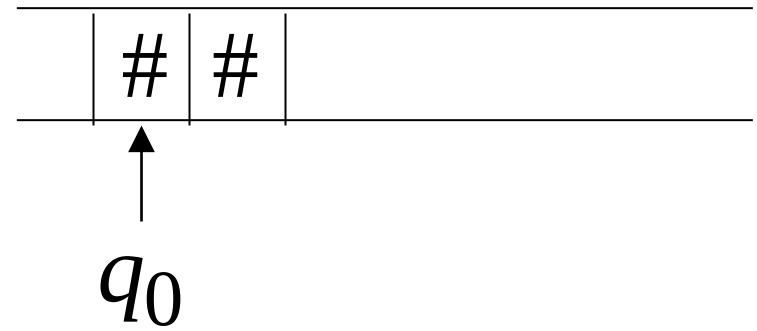
output $\rightarrow$ $s_1, s_2, s_3, \ldots$

Finite time: $t_1, t_2, t_3, \ldots$

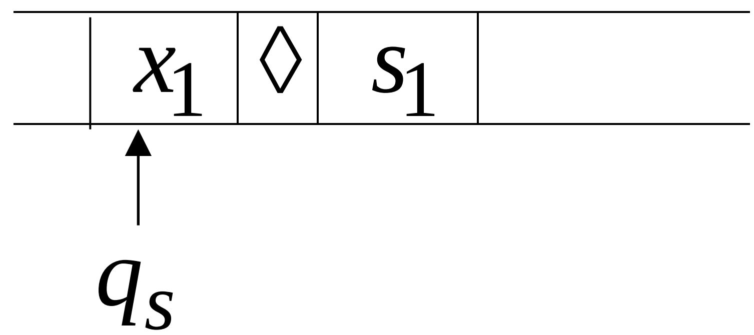
Enumeration Machine

Configuration

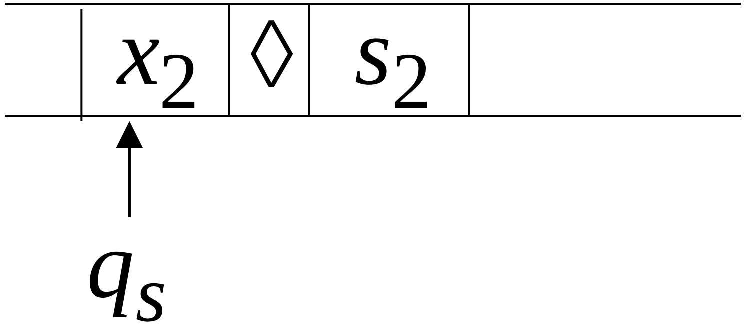
Time 0



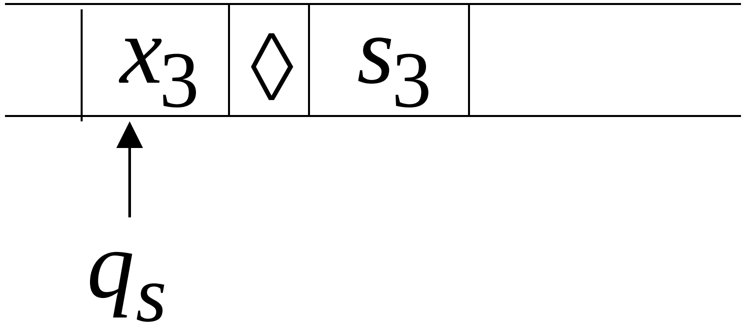
Time t_1



Time t_2



Time t_3



A set is countable if there is an enumeration procedure for it

Example:

The set of all strings $\{a,b,c\}^+$ is countable.

We will describe the enumeration procedure.

Naive procedure:

Produce the strings in lexicographic order:

$$a$$

$$aa$$

$$aaa$$

$$\ldots$$

Doesn’t work!

Strings starting with b will never be produced.

Better procedure: Proper Order

Produce all strings of length 1

Produce all strings of length 2

Produce all strings of length 2

....

Produce strings: a, b, c — Length 1

aa

ab

ac

ba

bb — Length 2

bc

Proper Order

ca

cb

cc

aaa

aab — Length 3

aac

....

Theorem:

The set of all Turing Machines
is countable.

Proof:

Any Turing machine is a finite string
Encoded with a sequence of 0's and 1's.

Find an enumeration procedure
for the set of Turing Machine strings.

Enumeration Procedure:

Repeat

1. Generate the next string of 0's and 1's in proper order

2. Check if the string defines a Turing Machine

    if YES: print string on output

    if NO: ignore string

Uncountable Sets

Definition:

A set is uncountable if it is not countable

Theorem:

Let S be an infinite countable set.

The powerset 2^S of S is uncountable.

The power set of natural numbers has the same cardinality as the set of real numbers. (Using the Cantor–Bernstein–Schröder theorem, it is easy to prove that there exists a bijection between the set of reals and the power set of the natural numbers).

Proof:

Since S is countable, we can write

$$S = \{s_1, s_2, s_3, \ldots\}$$

Element of S

Elements of the powerset have the form:

$$\{s_1, s_3\}$$

$$\{s_5, s_7, s_9, s_{10}\}$$

We encode each element of the power set with a string of 0's and 1's *

Powerset element	Encoding s_1	s_2	s_3	s_4	$\cdots$
$\{s_1\}$	1	0	0	0	$\cdots$
$\{s_2, s_3\}$	0	1	1	0	$\cdots$
$\{s_1, s_3, s_4\}$	1	0	1	1	$\cdots$
$\cdots$			$\cdots$		

*Cantor's diagonal argument

Let’s assume the contrary, that the powerset is countable.

We can enumerate the elements of the powerset.

Powerset element	Encoding					
t_1	1	0	0	0	0	$\cdots$
t_2	1	1	0	0	0	$\cdots$
t_3	1	1	0	1	0	$\cdots$
t_4	1	1	0	0	1	$\cdots$
$\cdots$	$\cdots$					

Take the powerset element whose bits are the complements of the diagonal.

t_1	1	0	0	0	0	$\cdots$
t_2	1	1	0	0	0	$\cdots$
t_3	1	1	0	1	0	$\cdots$
t_4	1	1	0	0	1	$\cdots$

New element: $0011\ldots$

(Diagonal complement)

The new element must be some t_i

This is impossible:

The i-th bit must be the complement of itself.

We have contradiction!

Therefore the powerset is uncountable.

Theorem:

Let S be an infinite countable set.

The powerset 2^S of S is uncountable.

Application: Languages

Alphabet: $\{a, b\}$

Set of Strings:

$$S = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinite and countable

Powerset: all languages

$$2^S = \{\{\lambda\}, \{a\}, \{a, b\} \{aa, ab, aab\}, \ldots\}$$

$$L_1 \quad L_2 \quad L_3 \quad L_4 \quad \cdots$$

uncountable

Languages: uncountable

L_1 L_2 L_3 $\cdots$ L_k $\cdots$

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

M_1 M_2 M_3 $?$

Turing machines: countable

There are infinitely many more languages than Turing machines!

There are some languages not accepted by Turing Machines.

These languages cannot be described by algorithms.

Recursively Enumerable Languages and Recursive Languages

Definition:

A language is **recursively enumerable** **if some Turing machine accepts it.**

Let L be a recursively enumerable language

and M be the Turing Machine that accepts it.

For a string w :

if $w \in L$ then M halts in a final state

if $w \notin L$ then M halts in some state

or loops forever

Definition:

A language is **recursive**
if some Turing machine accepts it
and halts on any input string.

In other words:

A language is recursive if there is
a membership algorithm for it

Let L be a recursive language

and M be the Turing Machine that accepts it.

For a string w :

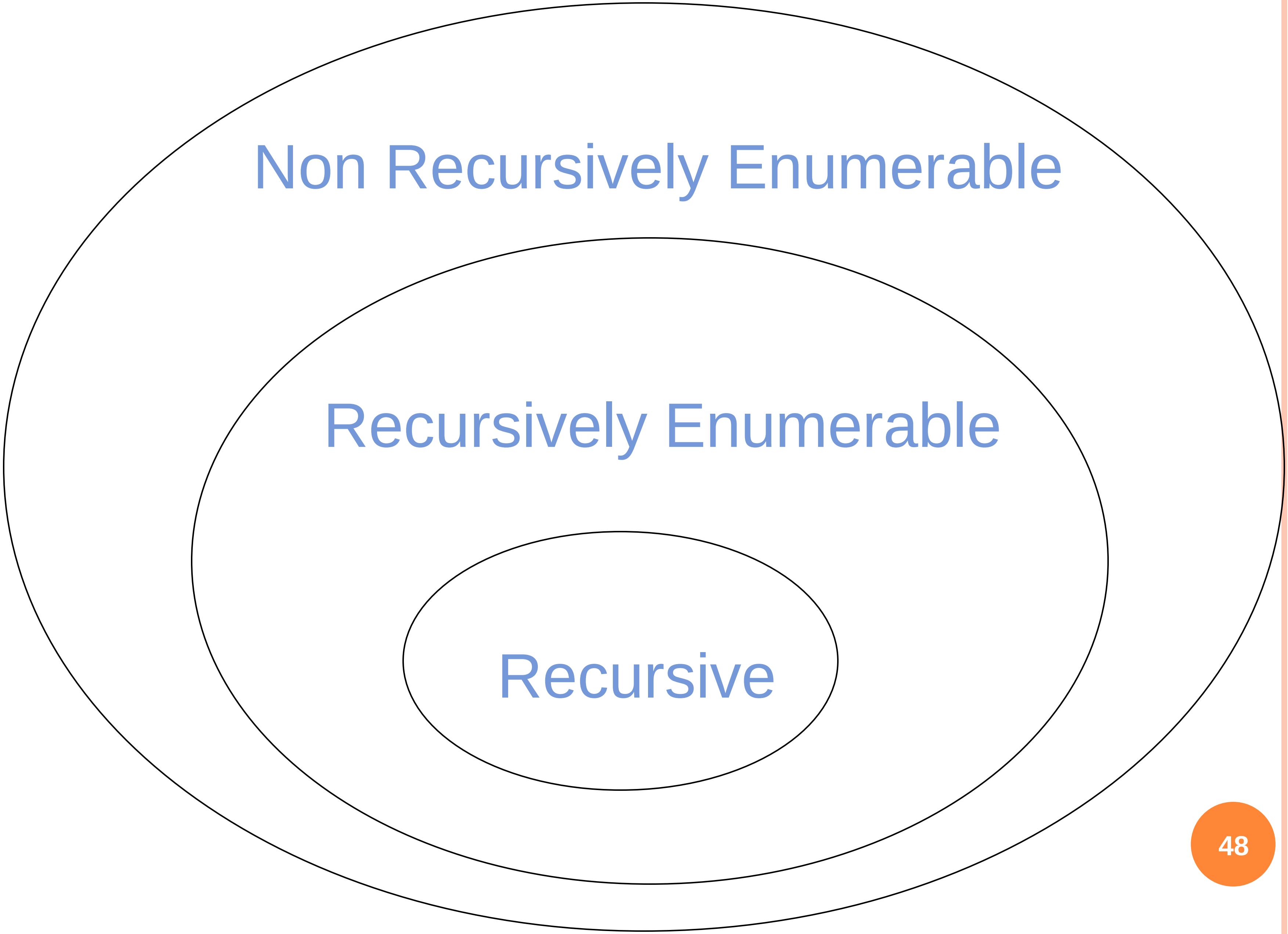
if $w \in L$ then M halts in a final state.

if $w \notin L$ then M halts in a non-final state.

We will prove:

1. There is a specific language which is not recursively enumerable.

2. There is a specific language which is recursively enumerable but not recursive.


Non Recursively Enumerable
Recursively Enumerable
Recursive

First we prove:

- If a language is recursive then there is an enumeration procedure for it.

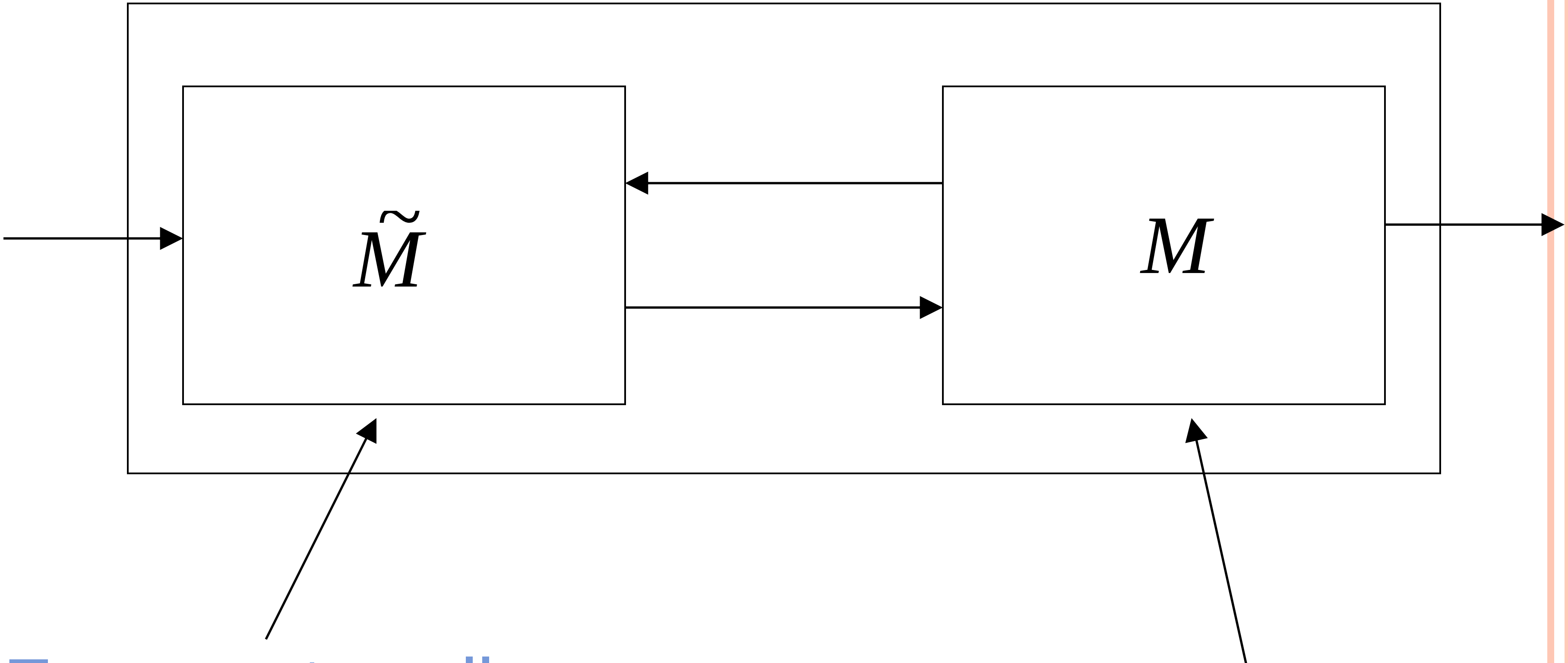
- A language is recursively enumerable if and only if there is an enumeration procedure for it.

Theorem:

if a language L is recursive then there is an enumeration procedure for it.

Proof:

Enumeration Machine



Enumerates all strings of input alphabet

Accepts L

Enumeration procedure

Repeat:

$\tilde{M}$ generates a string w

M checks if $w \in L$

YES: print w to output

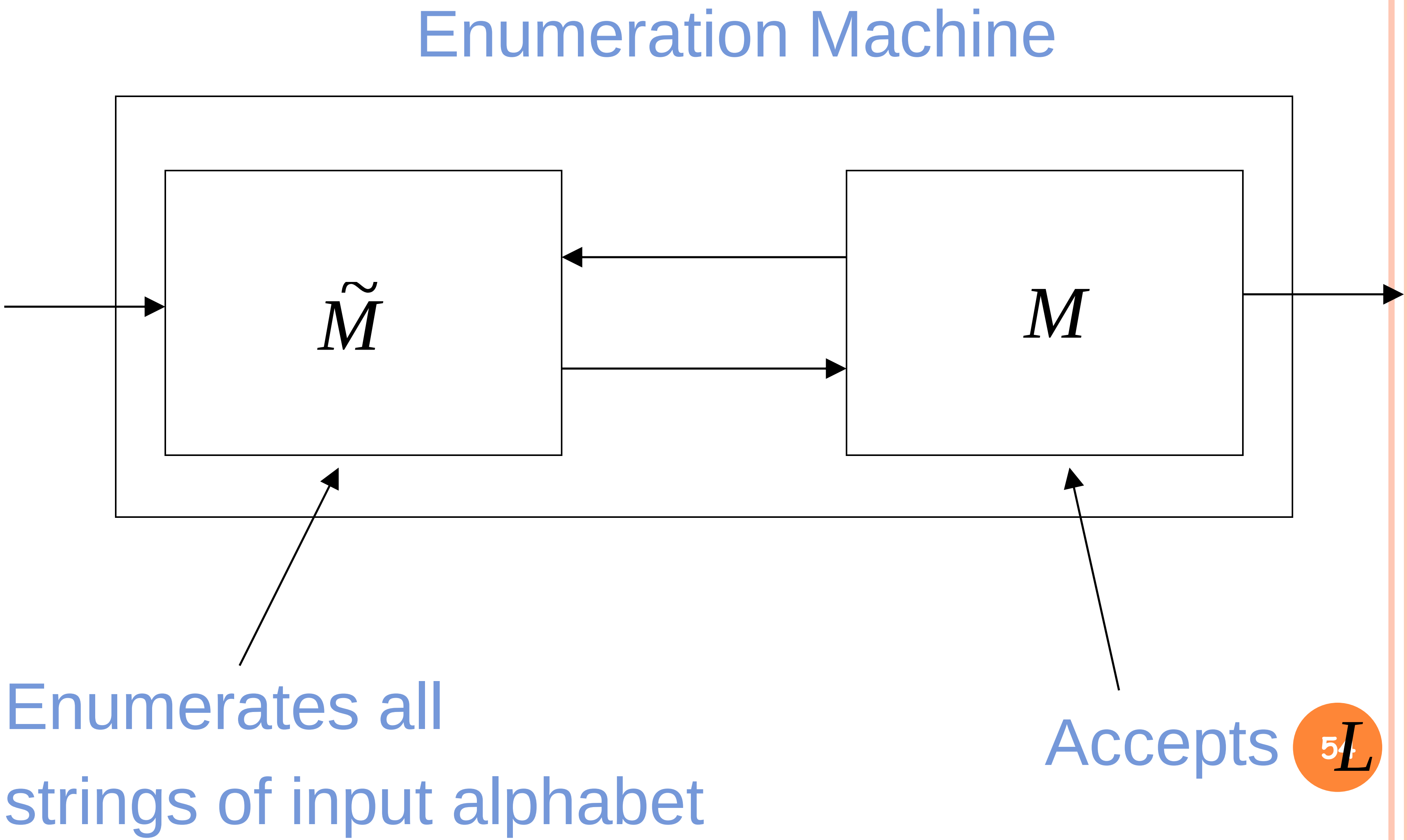
NO: ignore w

End of proof

Theorem:

if language L is recursively enumerable
then there is
an enumeration procedure for it.

Proof:



Enumeration Machine
$\tilde{M}$
M
Enumerates all strings of input alphabet
Accepts L

NAIVE APPROACH

Enumeration procedure

Repeat: $\tilde{M}$ generates a string w

M checks if $w \in L$

YES: print w to output

NO: ignore w

Problem: If $w \notin L$

machine M may loop forever

BETTER APPROACH

$\tilde{M}$ generates first string w_1

M executes first step on w_1

$\tilde{M}$ generates second string w_2

M executes first step on w_2

second step on w_1

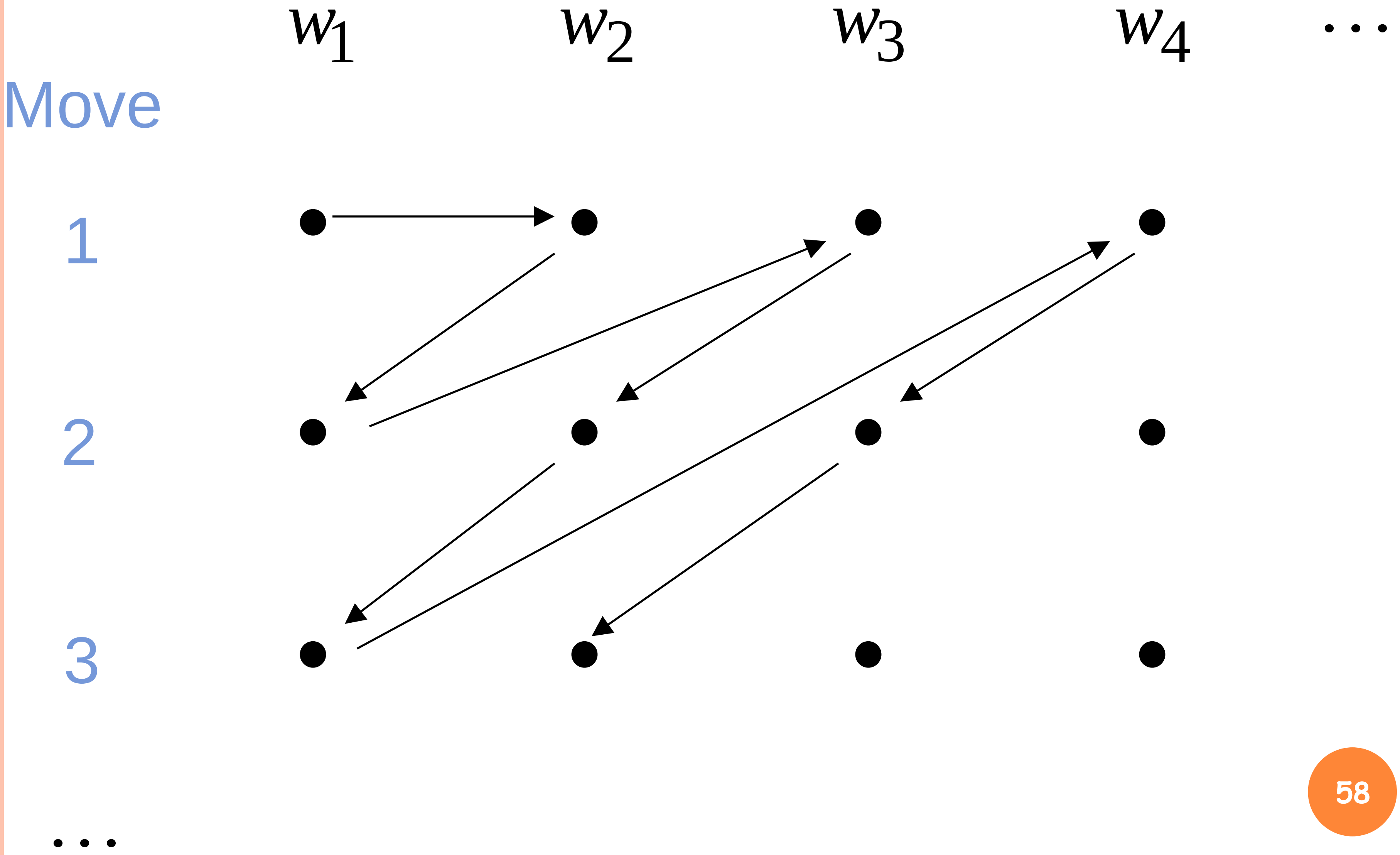
$\tilde{M}$ Generates third string w_3

M executes first step on w_3

second step on w_2

third step on w_1

And so on............



w_1
w_2
w_3
w_4
. . .
Move
1
2
3
. . .

If for string w

machine M halts in a final state

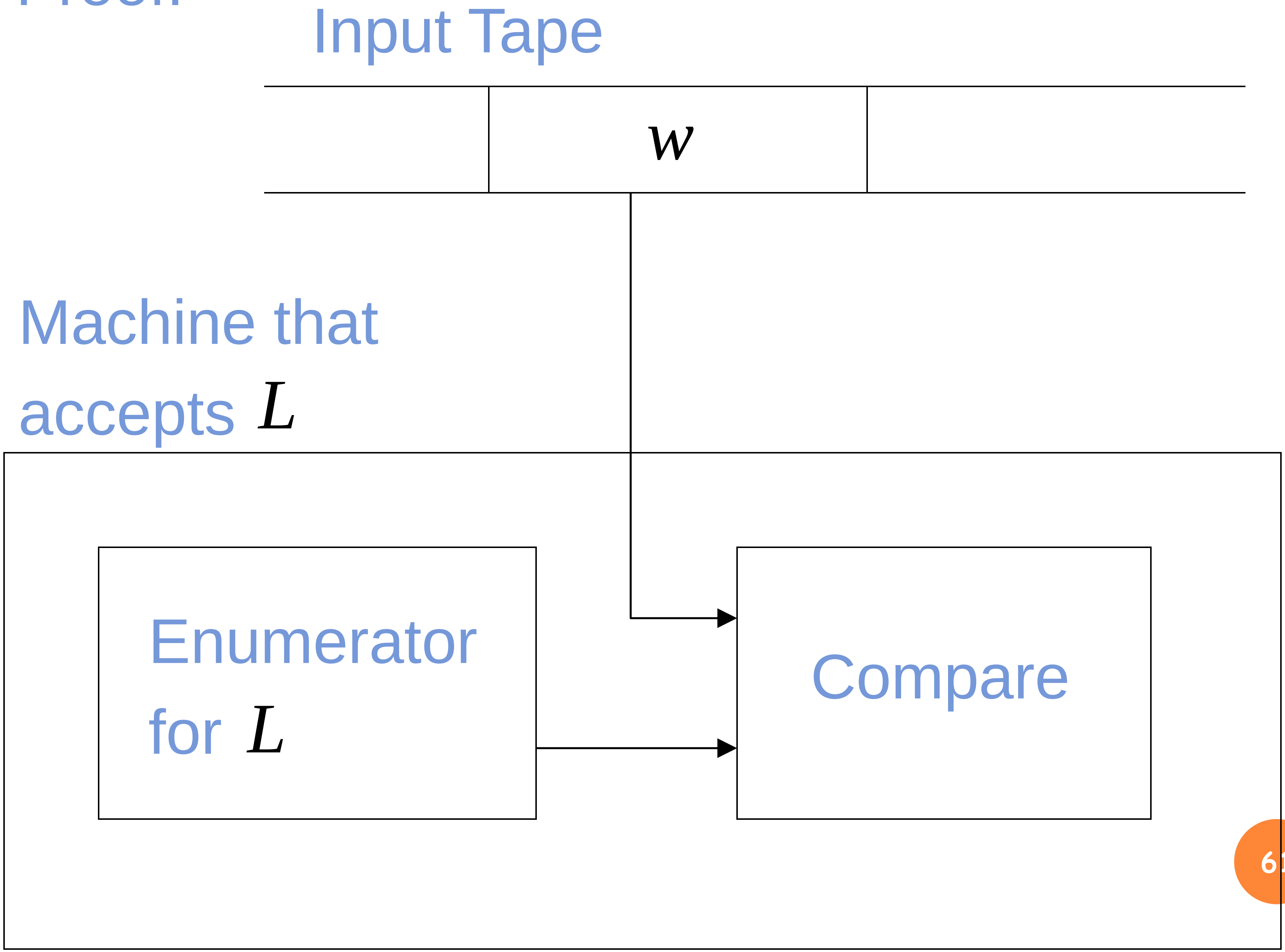
then it prints w on the output.

End of proof

Theorem:

If for language L
there is an enumeration procedure
then L is recursively enumerable.

Proof:



Input Tape
w
Machine that accepts L
Enumerator for L
Compare

Turing machine that accepts L

For input string w

Repeat:

- Using the enumerator, generate the next string of L

- Compare generated string with w

  If same, accept and exit loop

End of proof

Question:

This is not a membership algorithm.

Why?

Answer:

The enumeration procedure

may not produce strings in proper order

We have shown:

A language is recursively enumerable
if and only if
there is an enumeration procedure for it.

A Language which is not Recursively Enumerable

We search for a language that
is not Recursively Enumerable.

This language is not accepted by any Turing Machine.

Consider alphabet $\{a\}$

Strings: $a, \; aa, \; aaa, \; aaaa, \; \ldots$

$a^1 \quad a^2 \quad a^3 \quad a^4 \quad \ldots$

Consider Turing Machines
that accept languages over alphabet $\{a\}$

They are countable:

$$M_1,\ M_2,\ M_3,\ M_4,\ \ldots$$

Example language accepted by M_i

$$L(M_i) = \{aa, aaaa, aaaaaa\}$$

$$L(M_i) = \{a^2, a^4, a^6\}$$

Alternative representation

	a^1	a^2	a^3	a^4	a^5	a^6	a^7	$\cdots$
$L(M_i)$	0	1	0	1	0	1	0	$\cdots$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

Consider the language

$$L = \{a^i : a^i \in L(M_i)\}$$

L consists of the 1's on the diagonal

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L = \{a^3, a^4, \ldots\}$$

Consider the language $\overline{L}$

$$L = \{a^i : a^i \in L(M_i)\}$$

$$\overline{L} = \{a^i : a^i \notin L(M_i)\}$$

$\overline{L}$ consists from of 0’s on the diagonal

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$\overline{L} = \{a^1, a^2, \ldots\}$$

Theorem:

Language $\overline{L}$ is not recursively enumerable.

Proof:

Assume on the contrary that

$\overline{L}$ is recursively enumerable

There must exist some machine M_k that accepts $\overline{L}$

$$L(M_k) = \overline{L}$$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

Question: $M_k = M_1?$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$a^1 \in L(M_k)$$

Answer: $M_k \neq M_1$

$$a^1 \notin L(M_1)$$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	**0**	1	0	1	$\cdots$
$L(M_2)$	1	**0**	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

Question: $M_k = M_2$?

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$a^2 \in L(M_k)$$

Answer: $M_k \neq M_2$

$$a^2 \notin L(M_2)$$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

Question: $M_k = M_3$?

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$a^3 \notin L(M_k)$$

Answer: $M_k \neq M_3$

$$a^3 \in L(M_3)$$

Similarly: $M_k \neq M_i$ for any i

Because either:

$$a^i \in L(M_k) \quad a^i \notin L(M_i)$$

or

$$a^i \notin L(M_k) \quad a^i \in L(M_i)$$

Therefore the machine M_k cannot exist

CONTRADICTION!!!

The language $\overline{L}$

is not recursively enumerable.

End of proof

Observation:

There is no algorithm that describes $\overline{L}$

(otherwise it would be accepted by a Turing Machine)

A Language which is Recursively Enumerable and not Recursive

We want to find a language which

Is recursively enumerable

But not recursive

There is a Turing Machine that accepts the language

The machine doesn't necessarily halt on any input

We will prove that the language

$$L = \{a^i : a^i \in L(M_i)\}$$

Is recursively enumerable

but not recursive.

Theorem:

The language $L = \{a^i : a^i \in L(M_i)\}$

is recursively enumerable

Proof:

We will give a Turing Machine that accepts L

Turing Machine that accepts L

For any input string w

- Write $w = a^i$
- Find Turing machine M_i

  (using the enumeration procedure for Turing Machines)

- Simulate M_i on input a^i

- If M_i accepts, then accept w

End of proof

Observation:

Recursively enumerable

$$L = \{a^i : a^i \in L(M_i)\}$$

Not recursively enumerable

$$\overline{L} = \{a^i : a^i \notin L(M_i)\}$$

(Thus, not recursive)

Theorem:

The language $L = \{a^i : a^i \in L(M_i)\}$

is not recursive.

Proof:

Assume on the contrary that L is recursive.

Then $\overline{L}$ is recursive:

Take the Turing Machine M that accepts L

M halts on any input

If M accepts then reject

If M rejects then accept

Therefore:

$\overline{L}$ recursive

But we know:

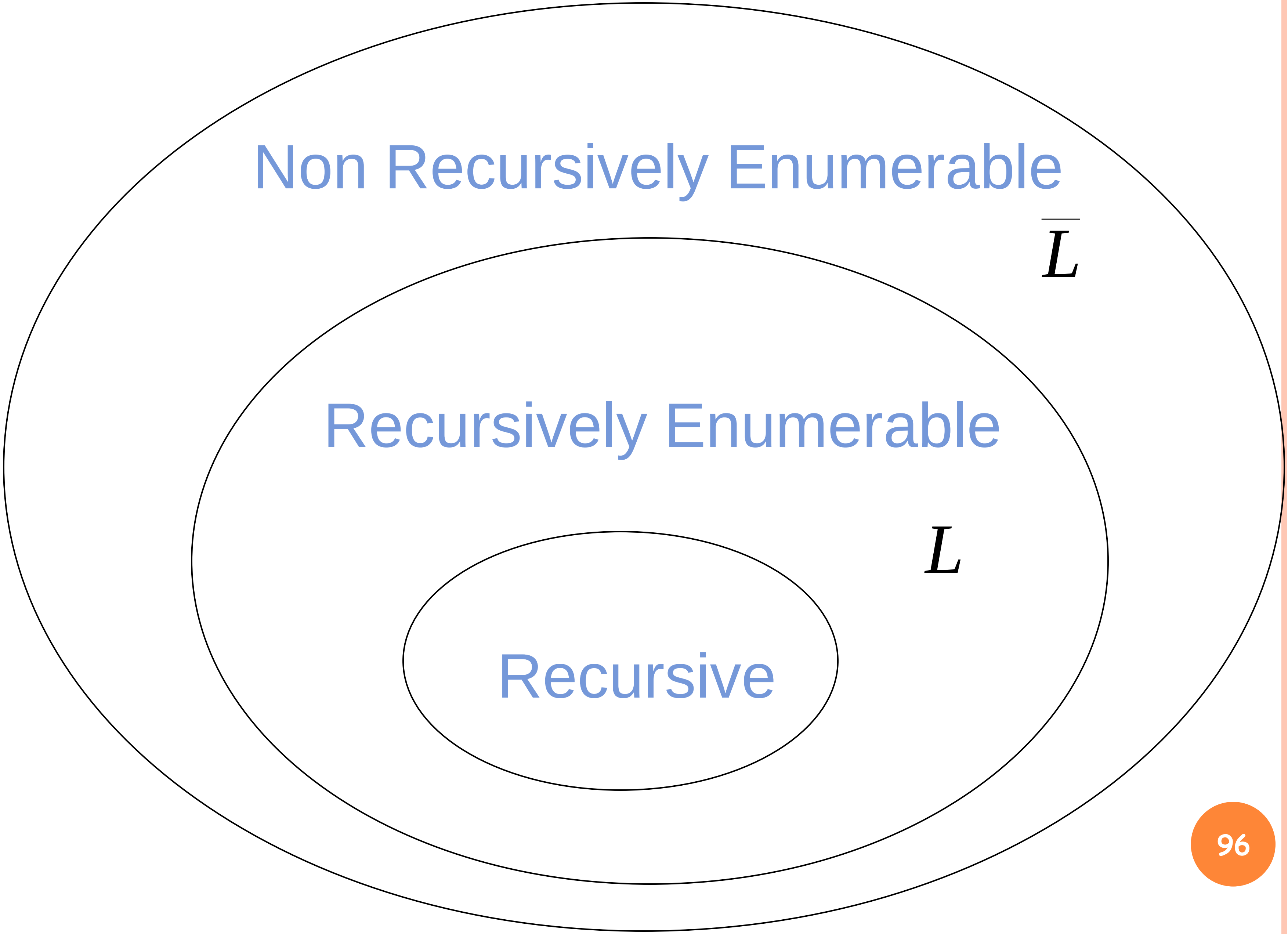
$\overline{L}$ not recursively enumerable

thus, not recursive

CONTRADICTION!

Therefore, L is not recursive

End of proof


Non Recursively Enumerable
$\overline{L}$
Recursively Enumerable
L
Recursive